

## **Remarks**

### **Introductory Comments**

Prior to this amendment, the present application included claims 67-72 and 78-100. Claims 1-66 and 73-77 were previously cancelled. With this amendment, Applicant amends the claims as reflected in the Listing of the Claims, cancels claims 89, 93, 97, 98 without prejudice or disclaimer of the subject matter thereof, and adds new claims 101-118. Thus, with this Amendment, claims 67-72, 78-88, 90-92, 94-96 and 99-118 are pending in the present application. Continued examination of the application, as amended, and in view of the following remarks, is respectfully requested.

### **Claim Rejections – 35 USC § 112, first paragraph**

Claims 67-72 and 78-87 were rejected under 35 USC § 112, first paragraph, as failing to comply with the written description requirement.

The MPEP states that: “An objective standard for determining compliance with the written description requirement is, “does the description clearly allow persons of ordinary skill in the art to recognize that he or she invented what is claimed.” (MPEP § 2163.02, first paragraph, quoting *In re Gosteli*, 872 F.2d 1008, 1012; 10 USPQ2d 1614, 1618 (Fed. Cir. 1989)). “The subject matter of a claim need not be described literally . . . in order for the disclosure to satisfy the description requirement.” (MPEP § 2163.02, third paragraph). “To comply with the written description requirement of 35 U.S.C. 112, para. 1 . . . each claim limitation must be expressly, implicitly, or inherently supported in the originally filed disclosure.” (MPEP § 2163.05).

“The analysis of whether the specification complies with the written description requirement . . . is conducted from the standpoint of one of skill in the art at the time the application was filed.” (MPEP § 2163.II.A.2) “The examiner has the initial burden of presenting evidence or reasoning to

explain why persons skilled in the art would not recognize in the original disclosure a description of the invention defined by the claims.” (MPEP § 2163.II.A.3(b)). “The fundamental factual inquiry is whether the specification conveys with reasonable clarity to those skilled in the art that, as of the filing date sought, applicant was in possession of the invention as now claimed.” (MPEP § 2163.I.B, last paragraph)

In Applicants’ response mailed March 7, 2006 (“Response”), responding to the Final Action, Applicants’ pointed out portions of the specification that would show to one of skill in the art that, as of the filing date, Applicants were in possession of the invention as claimed.

In the Advisory Action, the Examiner states that “there is a general mention of selecting and adding a silent guard variable to the software program in the Specification . . . However, none of the other portions of the Specification indicated by the Applicant illustrate the remaining steps . . . . All the portions Applicant points to . . . illustrates examples of a silent guard in use. These are two distinct inventions.” (Advisory Action, page 2, lines 3-9).

Applicants respectfully submit that Applicants’ specification would show to one of skill in the art that, as of the filing date, Applicants were in possession of the invention as claimed. , The Examiner finds that portions of the Specification pointed out in the Response “illustrates examples of a silent guard in use” (Advisory Action, page 2, lines 8-9). Applicants submit that, from examples of a silent guard in use and the accompanying explanation provided in the Specification, one of skill in the art would know Applicants were in possession of the invention as claimed at the time of filing. Thus, Applicants submit that the Specification meets the written description requirement of 35 U.S.C. § 112, first paragraph, and respectfully request that this rejection be withdrawn.

### **Claim Rejections – 35 USC § 102(a) and § 103(a)**

Claims 67-69, 71, 72, 78 and 80-100 were rejected under 35 USC § 102(a) as being anticipated by the Collberg article entitled “A Taxonomy of Obfuscation Transformations” (hereinafter “Collberg”). Claims 70 and 79 were rejected under 35 USC § 103(a) as being unpatentable over Collberg.

In the Advisory Action, the Examiner identifies an unintended loophole in Applicants’ remarks when trying to distinguish over Collberg using language similar to the Collberg definition. Applicants thank the Examiner for pointing out the loophole, and revise those arguments herebelow.

Collberg identifies 4 classes of technical program protection: *Obfuscation*, *Encryption*, *Server side execution* and *Trusted code* (Figure 1(a)), and then focuses almost entirely on obfuscation (Figure 1(b)-(g); Abstract, page 1, col. 1; and Introduction, page 2, col. 1, line 4 - col. 2, line 7). Collberg divides the targets of obfuscating transformations into 4 categories, *Layout obfuscation*, *Data obfuscation*, *Control obfuscation* and *Preventive transformation* (Figure 1(c)) and then presents a catalogue of obfuscating transformations for each of these categories (Figure 1(d)-(g)).

In the Final Action, the Examiner rejects claims 67-69, 71, 72, 78 and 80-100 based on Collberg, Section 7.1.3 entitled “Split Variable” (page 18) and the accompanying Figure 18 (page 19). Split variable is a type of data obfuscation transformation (Collberg, page 2, Figure 1(e); page 17, Section 7 entitled “Data Transformations”).

Collberg makes clear that the obfuscation techniques described in the paper “converts a program into an equivalent one that is more difficult to understand and reverse engineer.” (Collberg, page 1, col. 1, lines 12-14; *see also*; page 3, col. 2, lines 21-25, and page 4, col. 2,

lines 17-18).

As described in the Advisory Action and in Applicants' response to the Final Action, the data obfuscation transformations described in Section 7.1.3 of Collberg are one example of an obfuscating transformation from a source program P into a target program P' which Collberg defines on page 6, column 1, as:

- If P fails to terminate or terminates with an error condition, then P' may or may not terminate;
- Otherwise, P' must terminate and produce the same output as P.

The "Otherwise" portion of this definition can be rephrased in the positive as:

- If P terminates without an error condition, then P' must terminate and produce the same output as P.

This again emphasizes that, if the original program works (i.e., terminates without an error condition) then the transformed program must also work (i.e., terminate and produce the same output as the original program).

Applicants submit that Collberg not only does not teach or disclose the invention recited in Applicants' claims, but actually teaches away from the invention recited in the claims as explained below.

In order to expedite prosecution of the present application, many of the features patentably distinguishing the amended claims over Collberg are highlighted below. Additional features patentably distinguishing the amended claims over the prior art will be evident upon examination of the claims in this RCE.

#### Claim 67

Claim 67 recites a step of: "revising the selected computation to be dependent on the runtime

value of the silent guard variable, such that the software program executes improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable.” In contrast, Collberg requires that the target program terminates and produces the same output as the source program, i.e., that the target program executes properly. If “the software program executes improperly” as recited in claim 67, then it does not terminate and produce the same output as the source program as required by Collberg. The invention recited in claim 67 would be unacceptable under Collberg because even when the original program recited in claim 67 terminates properly, the program with the silent guard variable will not terminate properly if “the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable.” Thus, Collberg actually teaches away from Applicants’ invention as recited in claim 67. Accordingly, Applicants respectfully submit that claim 67 distinguishes over Collberg.

In addition, claim 67 recites “setting the runtime value of the silent guard variable to the expected value of the silent guard variable in the software program at a silent guard insertion point, the silent guard insertion point being separated from the execution point of the selected computation by a plurality of program instructions of the software program.” This limitation is disclosed in several places in the specification including page 32, lines 2-9 and page 86, lines 4-7. Collberg Figure 18(e) shows ten lines of code, and the variable the Examiner associates with the silent guard variable is variable “x” (Final Action, page 5, section 14.a). In Figure 18(e), Collberg does not show the statement that sets the value of variable “x” separated by a plurality of program instructions from the computation dependent on “x.” In fact, Collberg shows these statements in the same instruction line, namely lines (5’), (7’) and (8’). Thus, Collberg fails to teach “setting the runtime value of the silent guard variable to the expected value of the silent guard variable in the software program at a silent guard insertion point, the silent guard insertion point being separated from the execution point

of the selected computation by a plurality of program instructions of the software program” as recited in claim 67. Accordingly, Applicants respectfully submit that claim 67 distinguishes over Collberg.

For at least these reason, Applicants respectfully request that the Examiner withdraw the rejection and find claim 67 to be allowable over Collberg. Claims 68-72 are dependent on base claim 67. Accordingly, Applicants respectfully request that the Examiner find claims 67-72 allowable.

#### Claim 69

Claim 69 is dependent on claim 68 and base claim 67 and recites the further steps of “computing the runtime value of the silent guard variable using a mathematical expression including the runtime value of the selected program variable and the expected value of the selected program variable at the dependency point.” In the rejection of claim 69 in the Final Action, the Examiner points to Collberg, Fig. 18(e) steps 5’, 7’ and 8’ (Office Action, page 6, section 16), and in the rejection of base claim 67 the Examiner associates variable “x” with the silent guard variable (Office Action, page 5, section 14.a). The Examiner appears to be associating a1, a2, b1, b2, c1 and c2 with the program variable, though these variables are not in the source program which only has variables A, B and C. However, steps 5’, 7’ and 8’ only show the use of the runtime values of a1, a2, b1, b2, c1 and c2. Collberg has no teaching of a mathematical expression using both the runtime value and the expected value of a program variable as recited in claim 69. Collberg does not disclose, teach or suggest “computing the runtime value of the silent guard variable using a mathematical expression including the runtime value of the selected program variable and the expected value of the selected program variable at the dependency point” as recited in claim 69.

For at least this reason in addition to the reasons given with regard to claim 67, Applicants believe claim 69 is allowable over Collberg. Accordingly, Applicants respectfully request that the

Examiner so find and allow claim 69.

Claim 78

Claim 78 recites the steps of “revising the selected computation to be dependent on the runtime value of the program variable, such that the software program executes incorrectly if the runtime value of the program variable is not equal to the expected value of the program variable.” In contrast, Collberg requires that the target program terminates and produces the same output as the source program, i.e., that the target program executes correctly. If “the software program executes incorrectly” as recited in claim 78, then it does not terminate and produce the same output as the source program as required by Collberg. The invention recited in claim 78 would be unacceptable under Collberg because even when the original program recited in claim 78 terminates correctly, the program with the silent guard variable will not terminate correctly if “the runtime value of the program variable is not equal to the expected value of the program variable.” Thus, Collberg actually teaches away from Applicants’ invention as recited in claim 78. Accordingly, Applicants respectfully submit that claim 78 distinguishes over Collberg.

In addition, the Examiner associates the variables A, B and C of Collberg fig. 18(e) with the “program variable” of claim 78 (Final Action, page 9, section 24.e); and then for the step of “revising the selected computation to be dependent on the runtime value of the program variable, such that the software program executes incorrectly if the runtime value of the program variable is not equal to the expected value of the program variable,” the Examiner associates Collberg fig. 18(e) steps 1’-10’ (Final Action, page 10, section 24.h). The variables A, B and C of Collberg do not appear in any of steps 1’-10’ of Collberg. Thus, steps 1’-10’ are not dependent on the runtime values of variables A, B and C as required by claim 78. Accordingly, Applicants respectfully submit that claim 78 distinguishes over Collberg.

For at least these reasons, Applicants respectfully request that the Examiner withdraw the rejection and find claim 78 to be allowable over Collberg. Claims 79-81 are dependent on base claim 78. Accordingly, Applicants respectfully request that the Examiner find claims 78-81 allowable.

#### Claims 81

Claim 81 is dependent on base claim 78 and recites the further steps of “inserting a mathematical expression including the runtime value of the program variable and the expected value of the program variable into the selected computation.” In the rejection of claim 81, the Examiner points to Collberg, Fig. 18(e) steps 8’-10’ (Office Action, page 10, section 26), and in the rejection of base claim 78 the Examiner associates “variable A, B and C” with a program variable (Final Action, page 9, section 24.e). However, none of steps 8’-10’ show a mathematical expression using both runtime and expected values of a variables A, B or C, in fact none of variables A, B or C appear in steps 8’-10’. Collberg does not disclose, teach or suggest “inserting a mathematical expression including the runtime value of the program variable and the expected value of the program variable into the selected computation” as recited in claim 81.

For at least this reason in addition to the reasons given with regard to claim 78, Applicants believe claim 81 is allowable over Collberg. Accordingly, Applicants respectfully request that the Examiner so find and allow claim 81.

#### Claim 82

Claim 82 recites the steps of “selecting a program block containing a step necessary for proper execution of the software program, . . . selecting a silent guard for the program block; determining the expected value of the silent guard at the start of execution of the program block; and installing a branch instruction dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the branch



instruction causes the program block to be skipped, causing the software program to execute improperly.”

Collberg requires that if a program P terminates without an error condition, then the transformed program P' must terminate and produce the same output as P. (see Collberg, Section 4, page 6, col. 1 (definition of obfuscating transformation)). In contrast, the method of claim 82 recites that “if the runtime value of the silent guard is not equal to the expected value of the silent guard then the branch instruction causes the program block to be skipped, causing the software program to execute improperly.” Thus, in the method of claim 82, even if the original program terminates without an error condition, the program with guarding will skip “a program block containing a step necessary for proper execution of the software program” “if the runtime value of the silent guard is not equal to the expected value of the silent guard.” This violates the definition of obfuscating transformation given in Collberg. Thus, Collberg actually teaches away from Applicants' invention as recited in claim 82. Accordingly, Applicants respectfully submit that claim 82 is not anticipated by Collberg.

In addition, in the Examiner's rejection of claim 82, for the step of “installing a branch instruction dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the branch instruction will cause an incorrect branch to be taken,” the Examiner points to Collberg, page 18, section 7.1.3, especially 6<sup>th</sup> paragraph; and page 19, Fig. 18(e) steps 8'-10' after transformation. Steps 8' to 10' of Collberg Fig. 18(e) show the transformation of the IF-statements in steps 8-10 into equivalent IF-statements using the split variables. Collberg does not teach installing a branch instruction but simply translating an already existing branch instruction using split data variables. Collberg does not disclose “installing a branch instruction . . . in the software program, such that if the runtime value of

the silent guard is not equal to the expected value of the silent guard then the branch instruction causes an incorrect branch to be taken” as recited in claim 82.

For at least the reasons given above, Applicants respectfully request that the Examiner find claim 82 to be allowable over Collberg. Claims 83-87 are dependent on base claim 82. Accordingly, Applicants respectfully request that the Examiner find claims 82-87 allowable.

#### Claim 87

Claim 87 is dependent on base claim 82 and intervening claims 85 and 86. Claim 87 recites the further limitation that “the silent guard computation uses both the expected value of the program variable and the runtime value of the program variable.” In the rejection of claim 87, the Examiner points to Collberg, Fig. 18(e) steps 8’-10’ (Office Action, page 13, section 32). It appears from the rejection of intervening claim 85 that the Examiner associates variable “x” with the silent guard, and a1, a2, b1, b2, c1 and c2 with the program variable (Final Action, page 12, section 30.1). However, steps 8’-10’ do not show a computation that uses both the expected value and the runtime value of variables a1, a2, b1, b2, c1 or c2. Collberg does not disclose, teach or suggest a computation that “uses both the expected value of the program variable and the runtime value of the program variable” as recited in claim 87.

For at least this reason in addition to the reasons given with regard to claim 82, Applicants believe claim 87 is allowable over Collberg. Accordingly, Applicants respectfully request that the Examiner so find and allow claim 87.

#### Claim 88

Claim 88 recites “a program computation at an execution point . . . ; a silent guard variable having an expected value at the execution point; a program variable having an expected value at a dependency point . . . , the dependency point being separated from the execution point

by a plurality of program instructions, the runtime value of the silent guard variable being dependent on the runtime value of the program variable at the dependency point, . . . and wherein the program computation is dependent on the runtime value of the silent guard variable at the execution point, . . . the software program will execute improperly if the runtime value of the silent guard variable is not equal to the expected value of the silent guard variable at the execution point.” Thus, the dependency point where “the runtime value of the silent guard variable [is] dependent on the runtime value of the program variable” is “separated from the execution point” where “the program computation is dependent on the runtime value of the silent guard variable” “by a plurality of program instructions.” Collberg Figure 18(e) shows ten lines of code, and the variable the Examiner associates with the silent guard variable is variable “x” (Final Action, page 13, section 33). In Figure 18(e), Collberg does not show the statement that sets the value of variable “x” separated by a plurality of program instructions from the computation dependent on “x.” In fact, Collberg shows these statements in the same instruction line, namely lines 5’, 7’ and 8’. Thus, Collberg does not teach “the dependency point being separated from the execution point by a plurality of program instructions” as recited in claim 88. Accordingly, Applicants respectfully submit that claim 88 distinguishes over Collberg. Claim 90 depends on base claim 88. Accordingly, Applicants respectfully request that the Examiner find claims 88 and 90 allowable.

#### Claim 91

Claim 91 recites: “a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point; wherein the runtime value of the program variable is dependent on the result of the mathematical computation.” Collberg does not teach or disclose a

program variable that is dependent on a mathematical computation that includes both “the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable” as recited in claim 91. Accordingly, Applicants respectfully request that the Examiner find claim 91 allowable. Claims 92, 94 and 95 are dependent on base claim 91. Accordingly, Applicants respectfully request that the Examiner find claims 91, 92, 94 and 95 allowable.

Claim 96

Claim 96 recites “a program block . . . ; a silent guard in the software program having an expected value at the start of execution of the program block; a program variable having an expected value at an insertion point; the runtime value of the silent guard being dependent on the runtime value of the program variable at the insertion point . . . ; and a branch instruction dependent on the silent guard, the branch instruction being separated from the insertion point by a plurality of program instructions of the software program.”

Thus, the “branch instruction dependent on the silent guard” is separated “by a plurality of program instructions” from the insertion point where “the runtime value of the silent guard variable [is] dependent on the runtime value of the program variable.” In the rejection of claim 96, the Examiner associates “x” in Figure 18 with the silent guard variable, and seems to associate some combination of A, B, C, a1, a2, b1, b2, c1 and c2 with the program variable, though this is not clear (Final Action, page 16-17, section 41). In Figure 18, Collberg does not show a branch instruction dependent on “x” that is also separated by a plurality of instructions from where “x” is dependent on A, B, C, a1, a2, b1, b2, c1 or c2. Collberg does not teach “a program variable having an expected value at an insertion point; the runtime value of the silent guard being dependent on the runtime value of the program variable at the insertion point . . . ; and a branch instruction dependent on the silent guard,

the branch instruction being separated from the insertion point by a plurality of program instructions” as recited in claim 96. Accordingly, Applicants respectfully submit that claim 96 distinguishes over Collberg. Claims 99 and 100 are dependent on base claim 96. Accordingly, Applicants respectfully request that the Examiner find claims 96, 99 and 100 allowable.

#### **New Claims 101-118**

New claims 101-118 are added by this amendment as part of the RCE and are believed to be allowable over the prior art. New claims 101-105 are dependent on independent claims discussed above and add additional patentably distinguishing features to the respective claims. New independent claim 106 is directed to a recordable computer media having a tamper resistant software program recorded thereon, and claims 107 and 108 are dependent thereon. New independent claim 109 is also directed to a recordable computer media having a tamper resistant software program recorded thereon, and claims 110-113 are dependent thereon. New independent claim 114 is directed to a method for adding tamper resistance to a software program, and claims 115-118 are dependent thereon.

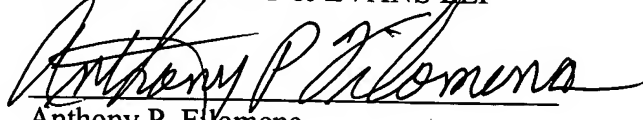
**Final Remarks**

Claims 67-72, 78-88, 90-92, 94-96 and 99-118 are pending in the present application and are believed to be in condition for allowance. Such allowance is respectfully requested.

In the event that there are any questions related to these amendments or to the application in general, the undersigned would appreciate the opportunity to address those questions directly in a telephone interview at 919-861-5092 to expedite the prosecution of this application for all concerned. If necessary, please consider this a Petition for Extension of Time to affect a timely response. Please charge any additional fees or credits to the account of Bose McKinney & Evans, LLP Deposit Account No. 02-3223.

Respectfully submitted,

BOSE McKINNEY & EVANS LLP

  
Anthony P. Filomena  
Reg. No. 44,108

Indianapolis, Indiana  
(317) 684-5000

697763\_1